Sentiment Analysis

Anghel Ana

Avram Ștefan Alexandru

Manolachi Tiberiu-Andrei

Voiculescu Ioana-Daria

Abstract

Sentiment analysis, a branch of natural language processing, plays a crucial role in understanding human emotions and opinions expressed in text data. In this study, we explore the effectiveness of various machine learning models including Logistic Regression, K-Nearest Neighbors (KNN), Naive Bayes, Decision Tree, and Support Vector Machine (SVM) for sentiment analysis tasks. The objective is to identify emotions conveyed in text data and compare the performance of different models in terms of accuracy.

1 Introduction

Our project is focused on developing a Natural Language Processing (NLP) model capable of automatically classifying the sentiment expressed in text data, specifically tweets, news articles, or reviews. This model will categorize sentiment as positive, negative, or neutral.

We were motivated towards this direction by the following aspects:

- **Automatization**: instead of manually analyzing large datasets (a process that is both expensive and time consuming), a NLP model automates the process, resulting in a faster and more efficient evaluation of public opinion;
- **Scalability**: the model could be applied to analyze gigantic amounts of data;
- **Decision making**: sentiment analysis allows businesses to measure customer satisfaction better, to understand brand perceptions, and to make informed decisions.

In the following sections, we will present our implementation and compare various approaches with the purpose of maximising our prediction accuracy.

2 Dataset

The dataset we chose consists of two CSV files: one `twitter.csv` file, with approximately 163 000 tweets, respectively a `reddit.csv` file containing around 37 000 comments. The data was extracted from Twitter and Reddit using their respective APIs (Tweepy for Twitter, PRAW for Reddit). The tweets and comments are related to Indian politics; more specifically, they are focused on opinions on Narendra Modi, other politicians and the opinions people have towards the next Prime Minister of India for the elections held in 2019. We used **only** the `twitter.csv` file, and we will refer to it from now on as *the dataset*.

The dataset has two columns:

- a column containing the **cleaned text data** (the tweets themselves);
- a second column of the **sentiment label** associated with each text entry - the labels are integers ranging from -1 to 1 (-1 = negative, 0 = neutral and 1 = positive);

We are not sure whether it was annotated manually or automatically, but, considering the dimension of the dataset, we believe the labels might have been generated automatically, probably by using a sentiment analysis tool.

Let us analyze how the data is distributed. The CSV file has 162 980 unique entries, where 35 509 tweets are negative (21.79%), 55 212 tweets are neutral (33.88%) and 72 249 positive (44.33%). Given the slant towards the positive in the dataset distribution, it was necessary to balance the data (we will present this in detail in 2.1).

2.1 Preprocessing

We used the following preprocessing methods:

2.1.1 Data balancing

We created a `read_data()` function in order to address the imbalance in the original data. The

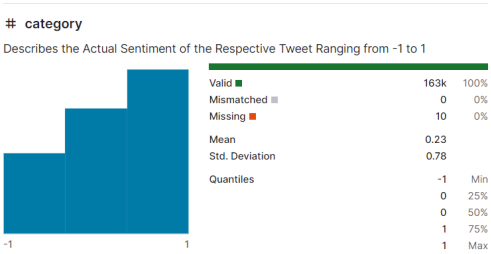

Figure 1: The distribution of our tweets data

function separates tweets based on sentiment (positive, negative, neutral), and randomly samples an equal number of tweets from each class to create a balanced dataset. In this way, we made sure that the model is trained on a more representative and uniform distribution of sentiments.

The `plot_data()` function shows a pie chart to represent the proportion of each category, helping us check if the data is really distributed evenly.

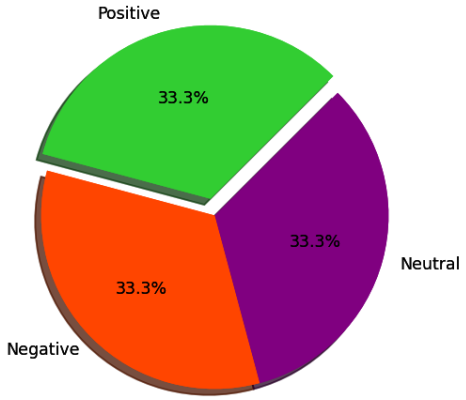

Figure 2: After balancing, the data is uniform

2.1.2 Text cleaning

Figuring that tweets are not clean text (as usually they have emojis or other kinds of symbols), the next step was to clean the data from the `"text"` column of the dataframe, using a `preprocessor()` function and the `BeautifulSoup` package. This function removes HTML tags, text-based emojis using regular expressions and unnecessary whitespaces, by joining words with a single space.

2.1.3 Stemming

At this step, we will reduce the words to their root forms, in order to improve generalization. We used `Porter Stemmer`, which works by iteratively applying rules to remove suffixes from words. After integrating it in a `apply_porter()` function, we used it in the TF-IDF vectorizer, which divides our text into individual units (words / phrases); they will be our **tokens**. In our case, the tokenizer will be `apply_porter()` function.

We continued with the **feature extraction** process, during which the vectorizer calculated a weight for each token, considering how often a word appears (Term Frequency) and how rare it is across the entire dataset (Inverse Document Frequency).

2.2 Models

We used different models (supervised machine learning models) like Logistic Regression, KNN, SVM, Decision Trees, Naive Bayes and even Bert.

Logistic Regression. Logistic regression analyzes the relationship between one or more independent variables and classifies data into discrete classes. The model estimates the mathematical probability of whether an instance belongs to a specific category or not. Our data is devided into positive, negative and neutral tweets. Logistic regression uses a logistic function called a sigmoid function to map predictions and their probabilities. The sigmoid function refers to an S-shaped curve that converts any real value to a range between 0 and 1. Logistic regression analysis yields reliable, robust, and valid results when a larger sample size of the dataset is considered.

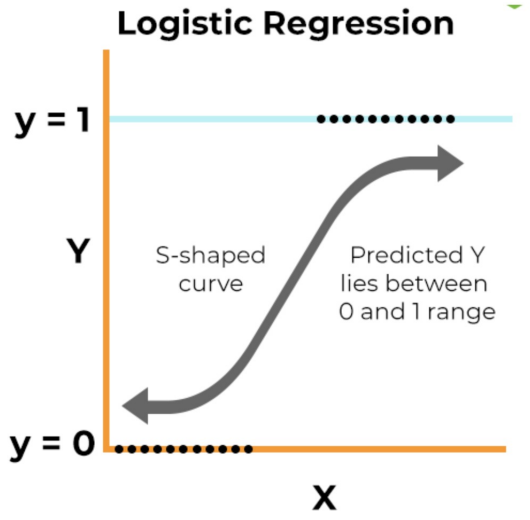

Figure 3: Sigmoid function

- cv: cross-validation—This means the training data will be divided into 6 subsets, and the model will be trained and evaluated 6 times,

using a different subset as the validation set each time.

- scoring: Specifies the evaluation metric.
- random state: reproducible if you run the code multiple times.
- max iter: maximum of iterations
- verbose: parameter that is linked to the information given at each iteration.

For cv=6, scoring= 'accuracy', random state=1, max iter=500, verbose=3 and jobs=4 we had 86.83% accuracy.

KNN k-nearest neighbors. This algorithm takes the k nearest neighbors based on a distance that is defined/ that we specify. (e.g. Euclidean distance) How it works: When a new data point is introduced, KNN finds its nearest neighbors in the feature space. These neighbors effectively define a region around the new point, which can be seen as a hyperplane separating the classes. The decision about the class of the new point is made based on the majority class of its nearest neighbors. If we visualize these neighbors in the feature space, they effectively form a hyperplane that delineates the boundary between different classes. At KNN we had the best accuracy when the number of neighbors was equal to 100 as well as the metric was "cosine".

Mean Test Score	Accuracy	Metric	Weights
0.5706	0.020041	Minkowski	Uniform
0.5716	0.020353	Minkowski	Distance
0.5714	0.020101	Cosine	Uniform
0.5750	0.019885	Cosine	Distance
0.5706	0.020041	Euclidean	Uniform
0.5716	0.020353	Euclidean	Distance

Hyperparameters:

1. weights: This can be either 'uniform' (all points in each neighborhood are weighted equally) or 'distance' (closer neighbors contribute more to the decision).
2. metric: This defines the distance metric used to calculate distances between points. The options here are 'minkowski', 'cosine', and 'euclidean'.

SVM (Support Vector Machine). A typical SVM separates these data points into red and black tags using the hyperplane, which is a two-dimensional line in this case. The hyperplane denotes the decision boundary line, wherein data points fall under the red or black category.

2.2.1 Hyperparameters:

- kernel:linear in our case(it's the way data it is transposed into the plane).
- random state: results of the model training will be reproducible

Naive Bayes. It is a generative model that models the joint probability distribution of the features and the labels. This contrasts with discriminative models like logistic regression, which model the decision boundary between classes. The "naive" aspect comes from the assumption that all features are independent given the class label. This assumption simplifies the computation of the probabilities and makes the algorithm very efficient. Naive Bayes is particularly effective in high-dimensional settings, such as text classification, where it can handle a large number of features without requiring a large amount of training data.

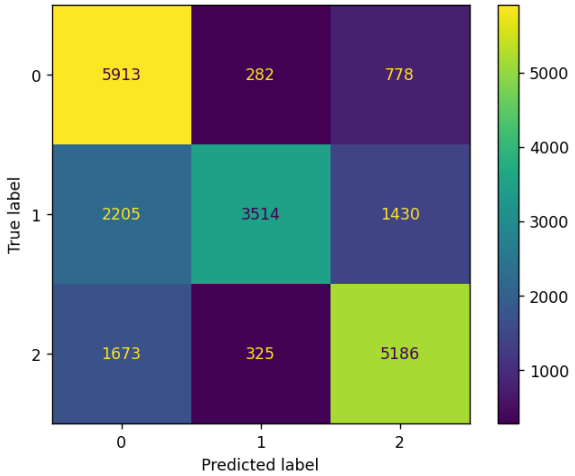

Figure 4: Confusion matrix

Decision Tree. Decision Trees (DTs) are non-parametric learning method used for regression and classification. The goal is to create a model that can predict the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation.

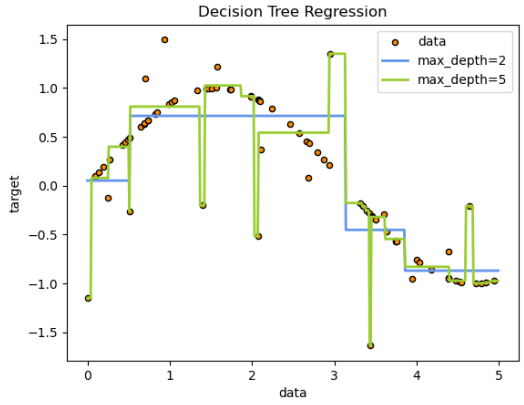

Figure 5: Approximate a sine curve with a set of if-then-else decision rules

3 Bert

We also tried to use BERT as a form of feature extraction. The tokenization done by BERT was attempting to occupy more memory then we had available and our trial was cut short. BERT can be used for classification task like sentiment analysis, the goal is to classify the text into different categories (positive/ negative/ neutral), BERT can be employed by adding a classification layer on the top of the Transformer output for the [CLS] token. The [CLS] token represents the aggregated information from the entire input sequence. This pooled representation can then be used as input for a classification layer to make predictions for the specific task.

4 Conclusion

After going through these five approaches, we can determine that the best model in terms of accuracy was Logistic Regresion (86.83% accuracy).

Model used	Accuracy
Logistic Regression	86.83%
SVM	77.65%
KNN	55.24%
Naive Bayes	68.59%
Decision Tree	78.02%

References

Twitter and reddit sentimental analysis dataset. https://www.kaggle.com/datasets/cosmos98/twitter-and-reddit-sentimental-analysis-dataset/data.

Munir Ahmad, Shabib Aftab, and Iftikhar Ali. 2017. Sentiment analysis of tweets using svm. *Int. J. Comput. Appl*, 177(5):25–29.

G Aliman, Tanya Faye S Nivera, Jensine Charmille A Olazo, Daisy Jane P Ramos, Chris Danielle B Sanchez, Timothy M Amado, Nilo M Arago, Romeo L Jorda Jr, Glenn C Virrey, and Ira C Valenzuela. 2022. Sentiment analysis using logistic regression. *Journal of Computational Innovations and Engineering Applications*, 7(1):35–40.

Mohammad Rezwanul Huq, Ali Ahmad, and Anika Rahman. 2017. Sentiment analysis on twitter data using knn and svm. *International Journal of Advanced Computer Science and Applications*, 8(6).

Xin Li, Lidong Bing, Wenxuan Zhang, and Wai Lam. 2019. Exploiting bert for end-to-end aspect-based sentiment analysis. *arXiv preprint arXiv:1910.00883*.